สรรค์ เสกขุนทด

วินโดวส์โปรแกรมมิง
ตอนที่ 1

เสนอแนวทางการพัฒนาโปรแกรมมิงบนวินโดวส์ เพื่ออำนวยความสะดวกและเพิ่มประสิทธิภาพการทำงานให้กับโปรแกรมเมอร์

สวัสดีครับท่านผู้อ่าน หลังจากที่ได้สัมผัสเรื่องราวของ GUIs ในคอลัมน์ร่ายยาว GUIs มาแล้ว ต่อไปจะเป็นการกล่าวถึงเรื่องการเขียนโปรแกรมบนไมโครซอฟต์วินโดวส์ ซึ่งในคอลัมน์นี้จะได้นำเสนอผู้อ่านตั้งแต่แนวความคิดเบื้องต้นของการพัฒนาแอพพลิเคชันบนไมโครซอฟต์วินโดวส์ (ต่อไปจะเรียกย่อ ๆ ว่าวินโดวส์) เทคนิคโปรแกรมมิงตลอดถึงการพัฒนาไลบรารีขึ้นมาใช้เอง และท่านจะได้ทราบว่าท่านก็สามารถเขียนโปรแกรมอย่าง aint brush, notepad หรือ clock ได้

พื้นฐาน

ก่อนที่จะลงมือเขียนโปรแกรมบนวินโดวส์ได้ ผู้อ่านจะต้องมีความคุ้นเคยกับการใช้ซอฟต์แวร์ไมโครซอฟต์วินโดวส์พอสมควร เช่น อาจจะเคยใช้ paint brush หรือ write เป็นต้น ระบบวินโดวส์ประกอบด้วยอุปกรณ์พื้นฐาน 4 อย่างที่เรียกว่า WIMP ได้แก่ วินโดวส์ (Window), ไอคอน (Icon), เมนู (Menu) และ อุปกรณ์ชี้ (Pointing device) ซึ่งในบทความนี้จะขอไม่กล่าวถึงเรื่องเหล่านี้อีก

สิ่งต่อไปที่จำเป็นต้องมีคือ ความสามารถในการเขียนโปรแกรมภาษา C หรือ C++ (ในกรณีที่ใช้ Borland C++) จริง ๆ แล้วเราสามารถพัฒนาด้วยภาษาปาสคาลหรือแอสเซมบลีก็ได้ แต่เพื่อความกะทัดรัด ผู้เขียนจะใช้ภาษา C เป็นหลักในการอธิบายโปรแกรมต่าง ๆ

แนวความคิดเบื้องต้น

ไมโครซอฟต์วินโดวส์เป็นระบบวินโดว์ ซึ่งสามารถรันได้หลายงานพร้อมกัน แต่เป็นระบบหลายงานในลักษณะที่เรียกว่า Non-preemtive นั่นคือ ความสามารถในการรันหลายงานต้องได้จากความร่วมมือของทุก ๆ แอพพลิเคชันที่รันในขณะนั้น ถึงตรงนี้ผู้อ่านอาจเกิดความสงสัยว่าร่วมมืออะไรและอย่างไร

การเขียนโปรแกรมบนวินโดวส์หรือบน GUIs อื่น ๆ ต่างจากการเขียนโปรแกรมบนดอสตรงที่ดอสมีลักษณะเป็นงานเดียว โปรแกรมที่รันอยู่จะครอบครองทรัพยากร (เช่น ซีพียู, เมมโมรี, พรินเตอร์, คีย์บอร์ด เป็นต้น) แต่เพียงผู้เดียว เมื่อโปรแกรมบนดอสรอรับคีย์บอร์ดจะไม่สามารถไปทำอย่างอื่นได้ แต่โปรแกรมบนวินโดวส์จะต้องแชร์ทรัพยากรเหล่านี้ร่วมกัน คือเมื่อโปรแกรมใช้ทรัพยากรจนเสร็จเรียบร้อยแล้ว ต้อง

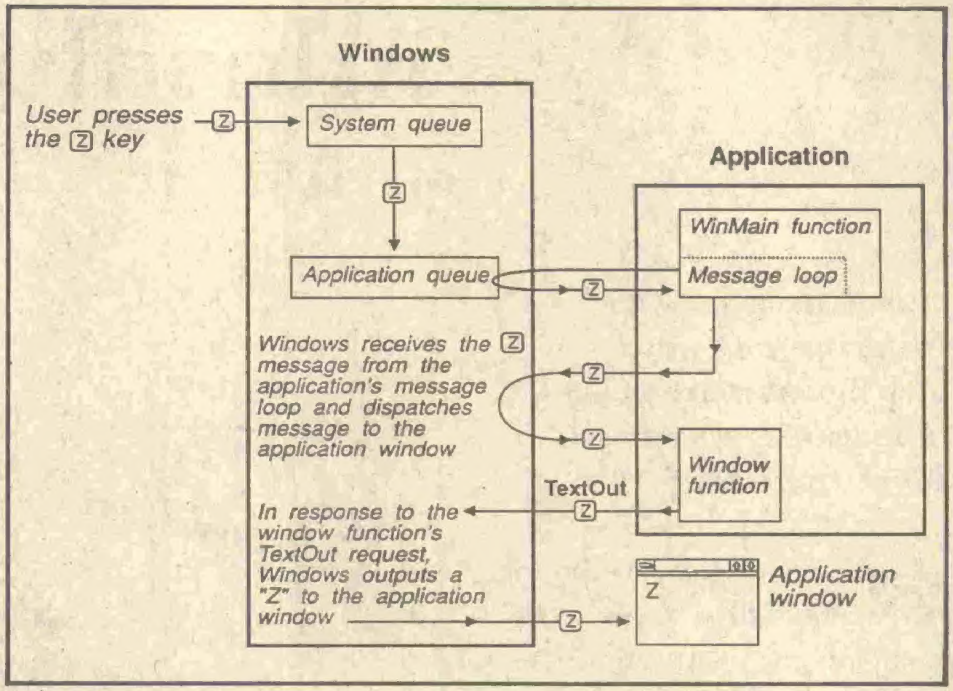
คืนทรัพยากรให้ระบบกลับคืนไป ผู้อ่านอาจจะสงสัยว่า แล้วจะเริ่มใช้เมื่อไหร่ จะคืนให้ระบบตอนไหน คำตอบอยู่ที่ Message

Message

Message คือข้อมูลข่าวสารที่บอกว่าเกิดอะไรขึ้นกับโปรแกรม โปรแกรมทำเพียงแค่คอยแมสเสจเหล่านั้น และเมื่อมีแมสเสจเข้ามา โปรแกรมก็จะตอบสนองต่อแมสเสจเหล่านั้นตามชนิดของแมสเสจ ที่มาของแมสเสจนั้นอาจจะมาจากวินโดวส์หรือมาจากโปรแกรมอื่น ๆ ก็ได้ โดยโปรแกรมสามารถส่งแมสเสจไปยังโปรแกรมอื่นได้

ตัวอย่างแมสเสจที่เห็นได้ชัดคือแมสเสจจากอุปกรณ์อินพุต เมื่อมีการคลิกเมาส์หรือกดคีย์บอร์ดขณะที่โปรแกรมกำลังรัน วินโดวส์จะจัดการข้อมูลที่มาจากเมาส์หรือคีย์บอร์ดและสร้างเป็นแมสเสจส่งมาให้โปรแกรมของเรา โปรแกรมก็จะตอบสนองตามชนิดของแมสเสจ เมื่อตอบสนองแล้วก็จะต้องกลับไปคอยแมสเสจอีก

ถึงจุดนี้ท่านผู้อ่านก็คงจะหายข้องใจแล้วว่า โปรแกรมจะต้องร่วมมือกันอย่างไรเพื่อให้เกิดระบบหลายงาน และเริ่มใช้ทรัพยากรเมื่อไรและคืนค่าตอนไหน คำตอบอยู่ที่การคอยแมสเสจนี้เอง วินโดวส์จะเอาช่วงเวลาที่โปรแกรมคอยแมสเสจนี้ไปทำอย่างอื่น เช่น ส่งแมสเสจไปยังโปรแกรมอื่น เมื่อโปรแกรมนั้นตอบสนองแมสเสจเสร็จเรียบร้อยแล้วก็จะคอยแมสเสจต่อไปอีก ถึงเวลานี้อาจจะเกิดแมสเสจของโปรแกรมอีก โปรแกรมก็จะตอบสนองต่อแมสเสจนั้นแล้วกลับไปคอยอีก เป็นอย่างนี้เรื่อยไป ซึ่งจะทำให้เป็นระบบหลายงานได้



รูปที่ 1 แสดงการประมวลผลแมสเสจ

จากการคอยแมสเสจนี้เอง คำถามที่ว่าโปรแกรมจะเริ่มทำงานเมื่อใด คำตอบคือเมื่อมีแมสเสจส่งเข้ามา เมื่อโปรแกรมได้รับแมสเสจก็จะทำการขอใช้ทรัพยากรจากระบบ และเมื่อทำงานเสร็จก็จะคืนทรัพยากรให้ระบบแล้วกลับอยู่ในสภาวะรอคอยแมสเสจ

Message Queue

แอพพลิเคชันทุกตัวจะมีสิ่งที่เรียกว่า message queue ซึ่งจะเป็นที่เก็บแมสเสจต่าง ๆ ที่ส่งมาให้แอพพลิเคชันนั้น ตามปกติแล้วแอพพลิเคชันจะได้รับแมสเสจจาก message queue ของตัวมันเองตามลำดับก่อนหลัง แต่จะมีบางกรณีที่แอพพลิเคชันจะได้รับแมสเสจโดยไม่ผ่าน message queue

วินโดวส์คืออะไร

ในมุมมองของโปรแกรมเมอร์แล้ว วินโดว์ (ไม่ใช่ "วินโดวส์" ซึ่งหมายถึงไมโครซอฟต์วินโดวส์) คือพื้นที่สี่เหลี่ยมบนจอที่แสดงข้อความต่าง ๆ และใช้รับอินพุต แอพพลิเคชันโดยส่วนใหญ่จะมีวินโดว์หลักและอาจจะมีวินโดว์ย่อย ๆ อีกก็ได้ วินโดว์ทุก ๆ วินโดว์จะมีวินโดว์ฟังก์ชัน (window function) ซึ่งจะเป็นฟังก์ชันที่คอยรับแมสเสจและตอบสนองต่อแมสเสจที่ส่งมาให้วินโดว์นั้น จากรูปที่ 1 เมื่อผู้ใช้กดคีย์บอร์ด แมสเสจจะเข้าไปที่ใน system queue และวินโดวส์จะนำแมสเสจไปวางไว้ที่ message queue ของแอพพลิเคชัน แอพพลิเคชันซึ่งกำลังคอยแมสเสจ (ใน message loop) ก็จะนำแมสเสจจาก queue ทำการส่งไปให้วินโดว์ฟังก์ชันเพื่อประมวลผลต่อไป ดังรูปที่ 1

เครื่องมือพัฒนาโปรแกรม

หลังจากที่ได้ทราบแนวความคิดเบื้องต้นแล้ว คราวนี้ก็มาดูว่าเราจะใช้อะไรในการสร้างโปรแกรม มีหลายบริษัท

ได้ผลิตทูลส์ (Tools) ในการพัฒนาโปรแกรมบนวินโดวส์ออกมา แต่ที่จะแนะนำในบทความนี้ได้แก่ SDK ของไมโครซอฟต์วินโดวส์ และ Borland C++ ของบอร์แลนด์

1. SDK (Software Development Kit)

ต้องการเนื้อที่บนฮาร์ดดิสก์ประมาณ 20 เมกะไบต์ ประกอบด้วย

1.1 คอมพายเลอร์ภาษาซี

1.2 โปรแกรมลิงค์

1.3 resource editor ได้แก่ SDK Paint, Dialog Editor และ Font Editor

1.4 resource compiler

นอกจากนี้ก็มีโปรแกรมสำหรับดีบักคือ CodeView สำหรับวินโดวส์, symdeb, Spy, swapping analyzer และ Heap Walker เป็นต้น ซึ่งในบทความนี้จะได้อธิบายต่อไป

2. Borland C++

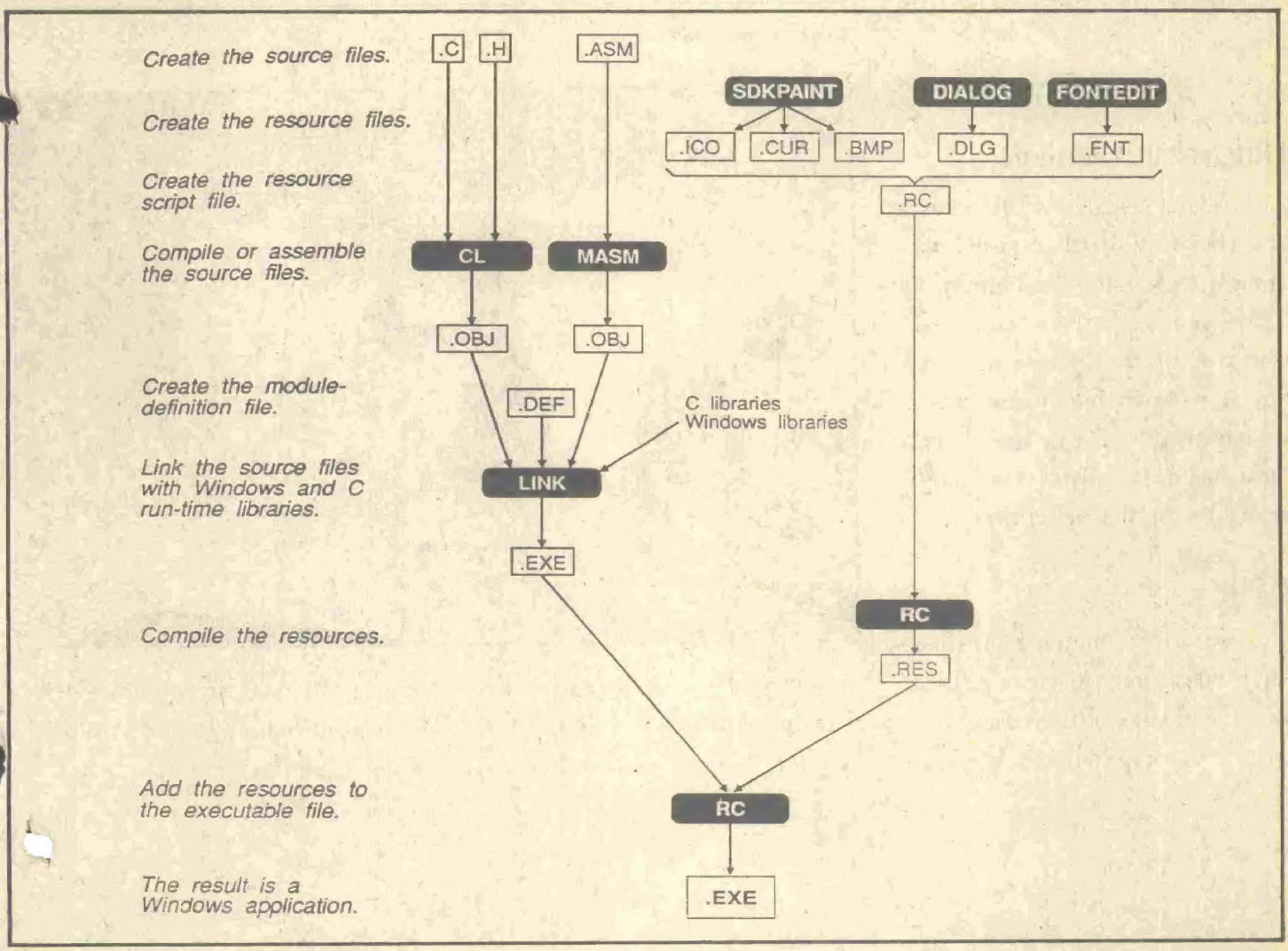
ประกอบด้วยโปรแกรมที่เรียกจากคอมมานด์ไลน์ ซึ่งเหมือนกับของไมโครซอฟต์ และโปรแกรมที่มีลักษณะเป็น integrated ซอฟต์แวร์ชุดนี้ต้องการเนื้อที่ประมาณ 15 เมกะไบต์ วิธีการใช้ตลอดจนความสะดวกก่อนข้างจะดีกว่าของไมโครซอฟต์ นอกจากนี้ยังสามารถใช้ C++ ได้อีกด้วย

ขั้นตอนการพัฒนาโปรแกรม

ในบทความนี้จะอิงไมโครซอฟต์เป็นหลัก สำหรับ Borland C++ ถ้าจะใช้คอมมานด์ไลน์ก็จะมีวิธีการใช้เหมือนของไมโครซอฟต์ ส่วน integrated จะคล้ายกับเทอร์โบซี

ขั้นตอนการพัฒนามีดังนี้ (ดังรูปที่ 2)

รูปที่ 2 ขั้นตอนการสร้างโปรแกรม



1. สร้างโปรแกรมภาษาซี (.C) ซึ่งประกอบด้วยส่วนที่จัดการแมสเสจต่าง ๆ

2. สร้าง resource script file (.RC) ซึ่งประกอบด้วย เมนู, บิตแม็พ, dialog, ฟอนต์ เป็นต้น

3. สร้างไฟล์ module-definition (.DEF) ซึ่งจะใช้สำหรับกำหนด เซกเมนต์, สแต็ก, ฮีพ เป็นต้น

4. คอมพายล์และลิงค์ไฟล์ภาษาซี (ได้ .EXE)

5. คอมพายล์ resource script file โดย resource compiler (ได้ .RES)

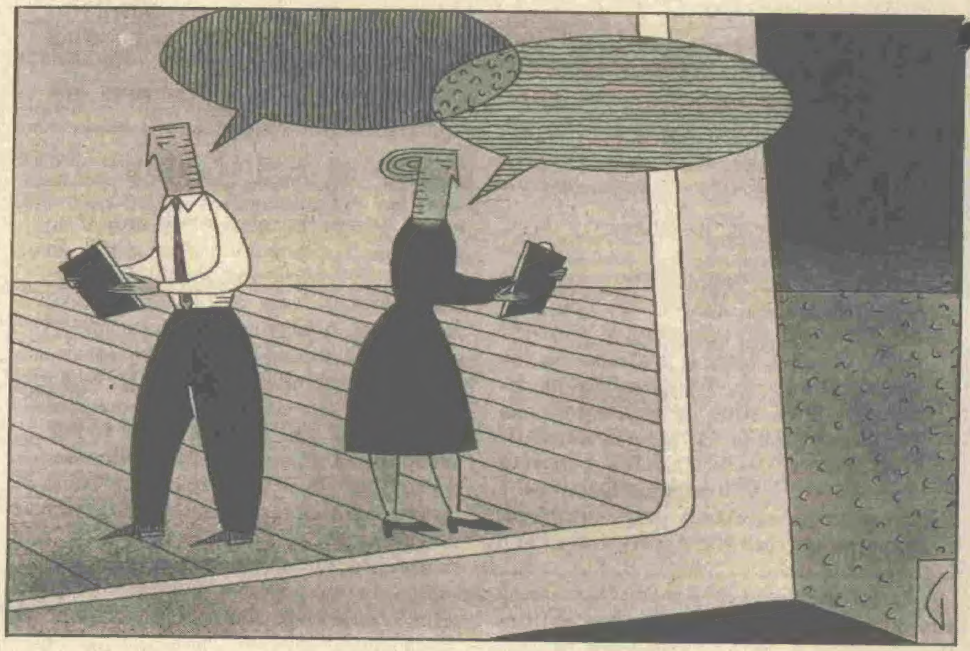
6. ใช้ resource compiler เพื่อนำ resource file (.RES) เพิ่มเข้าไปใน execute file

Hungarian Notation

โปรแกรมเมอร์ของไมโครซอฟต์จะมีสไตล์การเขียนโปรแกรมที่เรียกว่า Hungarian Notation คือตัวแปรทุกตัวจะมี prefix นำหน้าชื่อตัวแปร เพื่อบอกว่าตัวแปรนั้นเป็น type อะไร เช่น ตัวแปร wParam มีชื่อ Param, perfix w บอกว่าตัวแปรนี้มี type เป็น WORD (unsigned int) เป็นต้น ตารางที่ 1 เป็น prefix ต่าง ๆ ที่ใช้ในโปรแกรม

Prefix	*Data Type*
c	char
by	BYTE (unsigned char)
n	short or int
i	int
x, y	short (used as *x*-coordinate or *y*-coordinate)
cx, cy	short (used as *x* or *y* length; the c stands for "count")
b	BOOL (int)
w	WORD (unsigned int)
l	LONG (long)
dw	DWORD (unsigned long)
fn	function
s	string
sz	string terminated by 0 byte

ตารางที่ 1 แสดง prefix



สรุป

ผู้อ่านก็ได้ทราบแนวความคิดเบื้องต้นในการเขียนโปรแกรมบนวินโดวส์ และเครื่องมือที่จะใช้ในการพัฒนาโปรแกรม รวมทั้งวิธีการสร้างโปรแกรมอย่างคร่าว ๆ ไปพอสมควรแล้ว ในฉบับต่อไปจะเป็นโปรแกรม "hello window" ซึ่งจะเป็นโครงสร้างพื้นฐานของโปรแกรมต่อ ๆ ไป สำหรับฉบับนี้ก็มีเพียงแค่นี้ พบกันใหม่ฉบับหน้า สวัสดีครับ